Regridding with CDO

Introduction to Regridding

In order to **intercompare different models**, they must be interpolated from the native grid on which they were calculated to a common output grid. This procedure is known as **regridding or remapping**.

Fundamentally, **regridding consists of two steps**:

1. Creation of an interpolation matrix (weights file), which gives the amount that each point on the source grid contributes to each point on the target grid. These weights will depend on the type of interpolation desired. ISMIP6 has standardized on First-order Conservative Remapping (Jones, P.W. 1999, Monthly Weather Review, 127, 2204-2210), which does a better job preserving fluxes (or other integrals) than, eg., bilinear interpolation.
2. For each point on the target grid, sum the data values on the source grid multiplied by the corresponding weights. The same weights file may be used for more than one input variable, provided all are defined on the same source grid.

Different software may do one or the other or both of these steps.

To create the weights file, you must have *grid description files* for your source and target grids. These files give the latitudes and longitudes of every grid point, along with the corners of the boundary of its surrounding cell. There are several ways to write these files, depending on the software you're using.

We have tested a number of software packages, and so far, the easiest way we have found to do regridding for **structured** (Heiko Goelzer) and **unstructured triangular** (NASA Goddard) grids is using the Climate Data Operators (CDO)

Installing CDO

CDO (Climate Data Operator) claims to run on any POSIX-compatible Unix-type OS, including Mac OSX, most Linux, IBM AIX, HP-UX, Solaris, BSD variants, and even Cygwin. For those systems for which they're provided, by far the easiest way to install CDO is to use a package manager, since these will also automatically install any other libraries CDO needs (its dependencies). Available packages include MacPorts for OSX, .deb for Debian and Ubuntu, and .rpm for RedHat and Fedora. We have successfully installed CDO on OSX with MacPorts, but have not tested others (reports welcome!). Links to these packages are on the CDO Wiki here. To install via a package manager, you will probably need administrator privileges (this is certainly the case with MacPorts). For other systems, CDO must be built from source code. Instructions for doing so are also at the CDO web site here. We haven't tested this.

At the very least, you will also need the NetCDF library from Unidata so that CDO can read and write (classic) NetCDF files. If your model files are in NetCDF4 or HDF5 you'll also need the HDF5 library. The package managers should install these automatically if you don't already have them.

CDO Grid Description Files

The critical part of remapping with CDO is to create horizontal-grid description files describing both your input and output grids. These files provide the latitude and longitude of every grid point, and of the corners of its surrounding cell boundary.

A **CDO grid file** for the ISMIP6 standard 5 km polar stereographic grid for the Greenland ice sheet and the 8km polar stereographic grid for the Antarctic ice sheet can be obtained by emailing ismip6@gmail.com.

Details of the file format are in section 1.3 (especially 1.3.2.4) and **Appendix D** of the CDO User's Guide.

Here is a simple **example** of a grid that has **6 cells, 3 across and 2 up**:

```
# Simple 3x2 CDO grid file

gridtype = curvilinear
gridsize = 6
xsize = 3
ysize = 2

# Longitudes
xvals = 301.0  303.0  305.0
        301.0  303.0  305.0

# Longitudes of cell corners
xbounds =
302.0  302.0  300.0  300.0
304.0  304.0  302.0  302.0
306.0  306.0  304.0  304.0
302.0  302.0  300.0  300.0
304.0  304.0  302.0  302.0
306.0  306.0  304.0  304.0
```

```
# Latitudes
yvals = 61.0  61.0  61.0
        64.0  64.0  64.0

# Latitudes of cell corners
ybounds =
60.0  63.0  63.0  60.0
60.0  63.0  63.0  60.0
60.0  63.0  63.0  60.0
63.0  65.0  65.0  63.0
63.0  65.0  65.0  63.0
63.0  65.0  65.0  63.0
```

where:

gridtype = curvilinear

Several grid types are available. **Curvilinear** grids are 2-dimensional, but are not necessarily arranged along latitude and longitude lines. Because the standard polar-stereographic grid is rectangular in p-s x-y space but not lon-lat space, it is curvilinear. gridtype = unstructured is a simple list of grid points, with cells which may or may not be the same size, may be in any order, and may not necessarily be quadrilateral, and need not be adjoining. A Voroni triagulation is represented as an unstructured grid, as is an adaptive mesh. Notice that a curvilinear grid could as easily be described as unstructured, but the output variables would be 1D instead of 2D. For other options see the CDO User's Guide.

gridsize = 6 Total number of points in the grid. For input files, this MUST match the number of points in the variables. For curvilinear grids, gridsize = xsize * ysize. For unstructured grids, this is just the number of elements in your mesh.

xsize = 3 For curvilinear grids, the number of grid points in the longitude direction. Not used for unstructured grids.

ysize = 2 For curvilinear grids, the number of grid points in the latitude direction. Not used for unstructured grids.

nvertex = 4 Number of corners of the boundary of the cell surrounding each grid point. Optional for curvilinear grids (assumes 4), required for unstructured grids. A triangulation grid would have nvertex = 3.

xvals = ... The longitude position of every grid cell.

yvals = ... The latitude position of every grid cell.

xbounds = ... The longitudes of the corners of the cell boundary surrounding each grid point. There must be (nvertex * gridsize) values. CDO requires that the corners be listed counterclockwise around the grid point.

ybounds = ... The latitudes of the corners of the cell boundary surrounding each grid point. There must be (nvertex * gridsize) values.

Grid files may contain comment lines, which begin with “#”, and blank lines. In the example above, the grid point at (61.,302.) is the center of a cell with corners at (60.,302.), (63.,302.), (63.,300.), and (60.,300.).

Grid File Notes

- Each discrete **grid point** (xval,yval) can be considered representative of some region immediately surrounding it. This region constitutes the cell. (For example, the value at the grid point could be the mean of the value across the entire cell.) In regular grids the grid point is often at the center of the **cell**, although it doesn’t need to be. In order to conserve flux when remapping from one grid to another, it is necessary to know the area of each cell, so the cells’ boundaries must be described. This is done by specifying the position of each vertex along each cell boundary (xbounds,ybounds). CDO requires that cell-boundary vertices be given counterclockwise around the grid point.
- CDO identifies “x” with longitude and “y” with latitude (although there are options to rotate the pole). In curvilinear grids, cells are arranged with their longitudes (x) varying fastest, then latitudes (y). This matches the CF conventions. In **unstructured grids**, cells may be in any order.
- **Latitudes and longitudes** may have any number of decimal places. They may even be integers (assuming that is appropriate for your grid).
- In the grid files’ lists of latitudes and longitudes (xvals, yvals, xbounds, ybounds), values are separated by white space or newlines (not by commas). There can be any number of values on one line, however the total size of a line may not exceed 65,536 characters. (**Error** will be “readline Warning: end of line not found (maxlen = 65536)! Abort”) This can easily happen if you create the file using a word processor in which newlines are paragraph separators.

In the grid files, the keywords must be at the beginnings of their own lines. Something like this is **NOT** permitted:

```
# WRONG!
xvals = 300 301 302 yvals = 60 61 62
```

- The grid files can be large! The standard Greenland 5km grid is 20 MB, and a Greenland 1 km grid took 490 MB.
- CDO is also supposed to accept SCRIP format grid files, but we haven’t tested this.

Running the Remapping (Conservative Method)

ISMIP6 has standardized on First-order Conservative Remapping (Jones, P.W. 1999, Monthly Weather Review, 127, 2204-2210), which does a better job of preserving integrals of the data, like flux, between the source and target grids, although it may have a larger local interpolation error than other methods. The CDO operator remapcon implements this method, and both generates the interpolation weights matrix and applies the weights:

```
cdo remapcon,outgrid.txt -setgrid,ingrid.txt infile.nc outfile.nc
```

where:

infile.nc is the input netCDF file

outfile.nc is the output netCDF file

-setgrid,ingrid.txt (hyphen required) specifies the input grid description file. Note that this overrides any grid written in the input file. The total number of points (gridsize=) **MUST** match the number of points in the variables, but the structure (points in the x and y directions) can be different. (Formally, this runs the cdo setgrid operator to apply the grid description in ingrid.txt to infile.nc, before running the remapcon operator.)

remapcon,outgrid.txt says to do the conservative first-order remapping onto the grid described by outgrid.txt.

CDO’s syntax requires that there can be **NO** spaces following the commas after remapcon or setgrid.

remapcon is described in **chapter 2.12** of the CDO User’s Guide. There is further information about conservative remapping including an outline of the mathematical basis in the SCRIP User’s Guide.

By default all variables in the input netCDF file will be remapped. To only do some of them, add the cdo selname operator:

```
cdo remapcon,outgrid.txt -selname,var1,var2 -setgrid,ingrid.txt infile
.nc outfile.nc
```

which will select only the variables named var1 and var2 from the input file. You can specify as many as you like. See **section 2.3.2** of the CDO User's Guide for more options. As with -setgrid, there can be no spaces following the commas.

Practical Examples of CDO and NCO Tools

CDO is a suite of tools that enables the regridding of datasets as mentioned above. However, there is also a suite of tools called **NCO** (netCDF Operators). These tools can help prepare models for submissions by **enabling the user to add/remove/change variables, attributes** etc.

The easiest way to install the NCO tools is via **homebrew, macports, or anaconda'**. If that is not possible, the tools can be built from source via git. For more detailed information, consult the user guide here.

Below are some practical examples of these tools that may be of use to the community:

Do a conservative regridding of a grid from one resolution to another

Note that this is using the **remapYcon** for the regridding.

```
    # Names of the grid description files.  In this case the 10 & 5km
versions.
    ingrid="~/initMIP/grid_description_files/CDO/ISM_10km_CDO_file.txt
"
    outgrid="~initMIP/grid_description_files/CDO/ISM_05km_CDO_file.txt
"

    #names of input and output model datafiles
    infile="dlithkdt_GIS_BGC_BISICLES3_init_10km.nc"
    outfile="dlithkdt_GIS_BGC_BISICLES3_init_05km.nc"

    #Execute the regridding to 5km resolution
    cdo remapycon,$outgrid -setgrid,$ingrid $infile $outfile
```

Copy variables from one NetCDF file to another

This command copies the variables, “y, x, mapping” from: mapping_B03_05.nc to dlithkdt_GIS_BGC_BISICLES3_init_05km.nc

```
    xysrc="/mapping_B03_05.nc"
    outfile="dlithkdt_GIS_BGC_BISICLES3_init_05km.nc"

    ncks -A -v y,x,mapping $xysrc $outfile
```

Add an attribute to a variable in the NetCDF

This command operates on the file, file.nc. To this file, it will add an attribute called, grid_mapping to the variable, topg. The value of the attribute, *grid_mapping* will be mapping. -h means to not add this operation to the history of all operations performed on this file. -a specifies that this is an attribute o specifies that it is to be overwritten, so if there is already an attribute named, grid_mapping, it will be overwritten with the value, mapping.

c Specifies to create the attribute, grid_mapping if it does not exist already.

```
ncatted -h -a grid_mapping,topg,o,c,"mapping" file.nc
```

Remove unwanted Variables

Removes the variables, “lat_bnds,lon_bnds,lat,lon” and outputs a new file called, outfile.nc. -C if output file, outfile.nc, doesn’t exist then it will be created -O output to a new file. -x extract all variables from infile.nc and output to outfile.nc, **EXCEPT** for the variables specified -v controls which variables are being removed

```
ncks -C -O -x -v lat_bnds,lon_bnds,lat,lon infile.nc outfile.nc
```

Remove unwanted Attributes

This removes the attribute, history from the variable, global. This operation is performed on the file, file.nc

```
ncatted -h -a history,global,d,, file.nc
```

Rename a variable

This command will rename the variable, lat to latitude. It is operating on the file, file.nc

```
ncrename -v lat,latitude file.nc
```

Convert time from years to solar seconds

Assuming the file, infile.nc, has a time variable, and it is an array of years (e.g. 0,1,2,3,4,…). This will multiply the seconds per year by the year number.

```
    #1 year interval in seconds according to Heiko's grids - close to
"solar" time
    timeincr="31556926.0"

    timefactor='time=time*'$timeincr';'
    ncap2 -O -s $timefactor infile.nc outfile.nc
```

Convert Longitudes from E/W convention to 0-360 convention

This reads in the file, infile.nc, and if it has a variable named, lon it converts the negative longitude to a 0-360 notation.

```
ncap2 -O -s 'where(lon<0) lon=360+lon' infile.nc outfile.nc
```

Crop data to a certain geographic region

This command will crop data to just an area around Greenland. ncea still works, but it is being replaced by the command, nces. Note that the file must have variables, latitude and longitude.

```
ncea -d latitude,59.0,84.0 -d longitude,-95,-10 infile.nc outfile.nc
```

Regridding all variables in a model submission via a bash loop

This is an example of how to loop through all the variables of a group's model and do the regridding.

```
   #!/bin/bash

   #Names of the grid description files.  In this case the 10 & 5km ve
rsions
   ingrid="~/initMIP/grid_description_files/CDO/ISM_10km_CDO_file.txt"
   outgrid="~initMIP/grid_description_files/CDO/ISM_05km_CDO_file.txt"

   #grid for getting x,y, mapping variables for the "Bamber" 5km Green
land grid.
   xysrc="/~/initMIP/QGIS/mapping_B03_05.nc"

   #temporary files that get removed
   tmp1=$inpath"/tmp1.nc"

   #in this example, looking at the group, MIROC, and their ICIES00 ex
periment.
   ais=GIS
   agrp=MIROC
   amod=ICIES00
   #1 year interval in seconds - close to "solar" time
   timeincr="31556926.0"
   #------------------------------------------------------------------
----

   #get a list of all the variables.  Assumes variables are the same f
or
   #asmb, init, and ctrl directories.  you don't want the "scalar"
   #variable.
   vars=`ls $inpath/asmb|cut -d "_" -f 1`
   delete=(scalar)
   vars=${vars[@]/$delete}


   for amod in $amod;do
       for exp in asmb init ctrl; do

    #loop through each of the variables..
    for avar in $vars; do
       infile=${inpath}/${exp}/${avar}_${ais}_${agrp}_${amod}_${exp}.n
```

```
c
       outfile=${outpath}/${exp}/${avar}_${ais}_${agrp}_${amod}_${exp}
.nc

       echo $infile
       echo $outfile

       #First, the basic remapping operation.  This is the command
       #that does the actual regridding.  Note that it is writing it
       #to the temp file for now.
       cdo remapycon,$outgrid -setgrid,$ingrid $infile $tmp1

       #Now add in X,Y, and mapping variables from a file
       #that has a known good x,y grid, and the global Grid
       #attribute.  This operation copies the variables, "y, x, and
       #mapping" from Heiko's file to the temp file.  This helps the
       #file be read in programs like QGIS, Panoply, etc.
       ncks -A -v y,x,mapping $xysrc $tmp1

       #for QGIS to recognise the mapping variable we just added,
           #you need to add and attribute called, "grid_mapping" to
       #each data variable($avar)
       ncatted -h -a grid_mapping,$avar,o,c,"mapping" $tmp1

       #this converts all the variables to floats.  This helps save
       #A LOT of space, so is worth doing if double precision is not
       #really needed
       var2float=$avar'=float('$avar')'

    #-O does an overwrite
       ncap2 -O -s $var2float $tmp1 $tmp1

       #this removes unwanted variables.  Note writing out to
       #outfile name now
       ncks -C -O -x -v lat_bnds,lon_bnds,lat,lon $tmp1 $outfile

       #make adjustments to the mask variable's attributes....
       #space before/after brackets is essential!

       if [ ${avar} == "sfrgrf" ] || [ ${avar} == "sftflf" ] || [ ${av
ar} == "sftgif" ]; then
           ncatted -h -a units,$avar,o,c,"unitless" $outfile
              fi

       #add global attribute telling about interpolation
       ncatted -h -a Remapping,global,o,c,$avar" was conservatively re
```

```
gridded to a 5km grid using the 'cdo remapycon' utility. More details
on the remapycon utility can be found here: https://code.zmaw.de/proje
cts/cdo or by typing 'cdo -h remapycon' at the command prompt" $outfil
e

       #remove the history and other global attributes you may not wan
t:
       ncatted -h -a history,global,d,, $outfile
       ncatted -h -a CDI,global,d,, $outfile
       ncatted -h -a NCO,global,d,, $outfile
       ncatted -h -a CDO,global,d,, $outfile
       ncatted -h -a nco_openmp_thread_number,global,d,, $outfile
       ncatted -h -a history_of_appended_files,global,d,, $outfile

       rm -f $tmp1

          done
      done
   done
```

Notes

- It sounds from the CDO documentation as if it ought to be possible to read the input grid information directly from the input file, but we haven't gotten this to work yet. The **error** is "cdo remapcon (Abort): Unsupported grid type: generic". CDO may need particular NetCDF attributes to be in the file for this to work.
- CDO remapcon seems to handle multiple time steps just fine. However, the time dimension needs to be named time.
- The same horizontal grid must apply to all timesteps. There appears to be no way to have different horizontal grids at different timesteps. If your native grid changes during a simulation, you will therefore need to create a new grid description file for each time step where the mesh has changed.
- If the input file already has variables called "lon" and "lat", these will be remapped to the new grid as with all the other variables and renamed to "lon_2" and "lat_2". The output file will have lon and lat variables as defined by the output grid-description file.